(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 3 125 107 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
01.02.2017 Bulletin 2017/05

(51) Int Cl.:
G06F 3/12 *(2006.01)*

(21) Application number: **16187920.0**

(22) Date of filing: **02.02.2011**

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR

(30) Priority: **09.02.2010 US 302922 P**
09.02.2010 US 302916 P
04.06.2010 US 351466 P
04.06.2010 US 351461 P
24.06.2010 US 358306 P
31.08.2010 US 378827 P
31.08.2010 US 378832 P
14.09.2010 US 882116

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
11703991.7 / 2 534 573

(71) Applicant: **Apple Inc.**
Cupertino, CA 95014 (US)

(72) Inventors:
- **SWEET, Michael R.**
Cupertino, CA 95014 (US)
- **MILLER, Howard**
Cupertino, CA 95014 (US)

(74) Representative: **Rooney, John-Paul**
Withers & Rogers LLP
4 More London Riverside
London SE1 2AU (GB)

Remarks:
This application was filed on 08-09-2016 as a divisional application to the application mentioned under INID code 62.

(54) FRAMEWORK THAT SUPPORTS DRIVERLESS PRINTING

(57) The disclosed embodiments provide a printer (106) that facilitates driverless printing. This printer (106) includes a discovery component (212) configured to identify the printer (106) to a client (102), facilitate selection of the printer (106) by the client (102), obtain capability information from the printer (106) and send the capability information to the client (102). The printer (106) also includes a transport component (214) configured to receive printer data (208) from the client (102). Finally, the printer (106) includes a page-description-language component configured to print the received printer data (208).

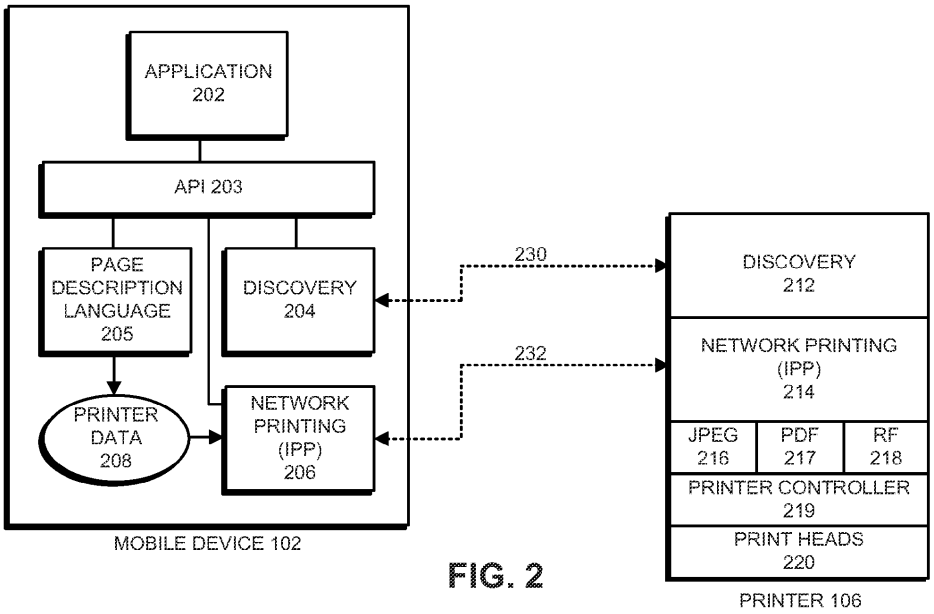

FIG. 2

EP 3 125 107 A1

Description

BACKGROUND

Field

[0001] The disclosed embodiments relate to printers for computer systems. More specifically, the disclosed embodiments relate to a computer-based printing system that operates without having to manage printer-specific driver software.

Related Art

[0002] Printers are often a problem for computer users. When a computer user initially installs a printer, the cabling and power cords are typically not a problem to hook up. However, the user typically has to install a printer-specific driver, which involves loading the driver from a disk or navigating to a website and downloading the driver. Even if the printer driver is already loaded into the computer system, the user often has to load and install an update for the driver from the printer manufacturer's website. These installation operations are time-consuming and commonly require the user to find and enter a long software-license key.

[0003] Hence, what is needed is a system that facilitates installing a printer without the above-described problems.

SUMMARY

[0004] The disclosed embodiments provide a system that facilitates driverless printing. This system includes a discovery component configured to identify accessible printers and to facilitate selection of an accessible printer, wherein the discovery component is also configured to obtain capability information from accessible printers. The system also includes a transport component configured to transport data to the selected printer, wherein the transport component is also configured to obtain capability information from the selected printer. Finally, the system includes a page-description-language component configured to generate printer data for the selected printer based on obtained capability information for the selected printer.

[0005] In some embodiments, the system provides an integrated framework that enables the discovery component, the transport component and the page-description-language component to be accessed by an application.

[0006] In some embodiments, the integrated framework includes one or more application programming interfaces (APIs) that provide access to the discovery component, the transport component and/or the page-description-language component. These APIs enable a calling application to interact with other application code being called through the API. Various function calls, messages or other types of invocations, which further may include various kinds of parameters, can be transferred via the API between the calling application and the code being called. In addition, the API may provide the calling application code the ability to use data types or classes defined in the API and implemented in the called application code. A method for operating through this API includes transferring one or more function calls, messages, other types of invocations or parameters via the API.

[0007] In some embodiments, while obtaining the capability information from accessible printers, the discovery component is configured to obtain current status information from the accessible printers.

[0008] In some embodiments, the current status information can specify: whether the printer is off-line; whether the printer is busy; or whether an error condition exists in the printer.

[0009] In some embodiments, the capability information for a printer can specify, for example, the file sizes supported by the printer; the file-format versions supported by the printer; the file-format extensions supported by the printer; the color spaces supported by the printer; the bit depths supported by the printer; or the resolutions supported by the printer.

[0010] In some embodiments, the capability information obtained from the selected printer enables the system to generate printer data for the selected printer without the need for the system to maintain printer-specific software or printer-specific configuration information for the selected printer.

[0011] Some embodiments include one or more application programming interfaces (APIs) in an environment with calling program code that interacts with other program code being called through the one or more interfaces. Various function calls, messages or other types of invocations, which further may include various kinds of parameters, can be transferred via the APIs between the calling program and the code being called. In addition, an API may provide the calling program code the ability to use data types or classes defined within the API and implemented in the called program code.

[0012] At least certain embodiments include an environment with a calling software component interacting with a called software component through an API. A method for operating through an API in this environment includes transferring one or more function calls, messages, and other types of invocations or parameters via the API.

BRIEF DESCRIPTION OF THE FIGURES

[0013]

FIG. 1 illustrates a printing system in accordance with the disclosed embodiments.

FIG. 2 illustrates components involved in performing a printing operation in accordance with the disclosed embodiments.

FIG. 3 presents a flow chart illustrating the printing

process in accordance with the disclosed embodiments.

FIG. 4 presents a flow chart illustrating the driverless printing process in accordance with the disclosed embodiments.

FIG. 5 presents a flow chart illustrating the printing process from the printer's perspective in accordance with the disclosed embodiments.

FIG. 6 illustrates the structure of a universal-raster-format-supported (URF-supported) key in accordance with the disclosed embodiments.

FIG. 7 illustrates the structure of a device-independent bitmap container in accordance with the disclosed embodiments.

FIG. 8 illustrates how software components call each other through an API in accordance with the disclosed embodiments.

FIG. 9 illustrates how API calls can be made through a system stack in accordance with the disclosed embodiments.

DETAILED DESCRIPTION

[0014] The following description is presented to enable any person skilled in the art to make and use the disclosed embodiments, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the disclosed embodiments. Thus, the disclosed embodiments are not limited to the embodiments shown, but arc to be accorded the widest scope consistent with the principles and features disclosed herein.

[0015] The data structures and code described in this detailed description are typically stored on a non-transitory computer-readable storage medium, which may be any device or medium that can store code and/or data for use by a computer system. The non-transitory computer-readable storage medium includes, but is not limited to, volatile memory, non-volatile memory, magnetic and optical storage devices such as disk drives, magnetic tape, CDs (compact discs), DVDs (digital versatile discs or digital video discs), or other media capable of storing code and/or data now known or later developed.

[0016] The methods and processes described in the detailed description section can be embodied as code and/or data, which can be stored in a non-transitory computer-readable storage medium as described above. When a computer system reads and executes the code and/or data stored on the non-transitory computer-readable storage medium, the computer system performs the methods and processes embodied as data structures and code and stored within the non-transitory computer-readable storage medium. Furthermore, the methods and processes described below can be included in hardware modules. For example, the hardware modules can include, but are not limited to, application-specific integrated circuit (ASIC) chips, field-programmable gate arrays (FPGAs), and other programmable-logic devices now known or later developed. When the hardware modules are activated, the hardware modules perform the methods and processes included within the hardware modules.

Driverless Printing

[0017] The disclosed embodiments facilitate "driverless printing," which enables a computing device to print to a printer without having to worry about installing or updating associated printer driver software. This is accomplished by augmenting a discovery protocol (such as Bonjour™) and an associated network-printing protocol (such as IPP), so that the protocols return printer capability information which is used to generate printer data for a selected printer.

[0018] Note that the disclosed embodiments differ from existing systems which need to maintain specific drivers for each supported printer. In these existing systems, the features and capabilities for specific printers arc stored in static capability files created by the printer vendors, such as a PostScript Printer Description (PPD) file, and these static capability files are stored on the mobile device itself. In contrast, the disclosed embodiments query a printer (not a static file) to determine the "current" operational capabilities of a printer. In addition to the discovery protocol and the network-printing protocol, the disclosed embodiments also provide a page-description language that specifies the appearance of printed pages.

[0019] The disclosed embodiments additionally provide a framework which enables an application to easily use the discovery protocol, the network-printing protocol and the page-description language. This framework can be implemented using one or more APIs which enable an application to access these components. Details of the disclosed embodiments are described below, but first we describe the printing system.

Printing System

[0020] FIG. 1 illustrates a printing system 100 in accordance with the disclosed embodiments. Printing system 100 includes a computing device 102 and a printer 106. Computing device 102 can generally include any type of computer system or computing device, such as a personal computer system, a server computer system, a laptop computer, a notebook computer, a tablet computer, a personal digital assistant, a digital media receiver (such as Apple TV™), a digital picture frame, a cellular telephone or a portable navigation system. Printer 106 can generally include any device or system capable of printing textual data or images onto some type of print media, such as paper or photo media. For example, printer 106 can comprise a stand-alone printer, or alternatively a printing system, which includes a print server that is

coupled to one or more printers.

[0021] Computing device 102 is coupled to printer 106 through network 104. Network 104 can generally include any type of communication channel capable of coupling together network nodes. For example, network 104 can include a wireless network connection, such as a Bluetooth™ network connection; a cellular networking connection (e.g., a 3G/4G network or an Edge network); a networking connection based on the standards described in Institute for Electrical and Electronic Engineers (IEEE) 802.11; a wireless personal-area networking (PAN) connection, such as a network connection based on the standards described in IEEE 802.15; or any peer-to-peer (wireless or wired) networking technology. Network 104 can also include a wired network connection, such as a network connection based on the standards described in IEEE 802.3.

[0022] During the printing process, computing device 102 initially makes contact with printer 106 through a discovery protocol as is described in more detail below. Next, after printer 106 is identified and selected, there arc a number of ways that printing can be accomplished. If computing device 102 possesses a driver for printer 106, or if printer 106 supports driverless printing, computing device 102 can use the driver or the driverless printing protocol to generate printer data for a print job. The printer data can then be sent to printer 106 through network 104. This process is described in more detail below. (Note that the term "driverless printing" refers to a printing technique which operates without the need for printer-specific software or printer-specific configuration on a computing device.)

Printing Components

[0023] FIG. 2 illustrates components involved in performing a printing operation in accordance with the disclosed embodiments. Referring to FIG. 2, computing device 102 includes an application 202 which needs to perform a printing operation. For example, application 202 can include a printing utility that allows a user to print a specific file. Alternatively, application 202 can include any type of general-purpose or special-purpose computer-based application that periodically needs to perform a printing operation, such as a word processor. Application 202 makes calls to API 203, which provides a framework that facilitates access to a number of components, including a discovery component 204, a page-description-language component 205 and a network-printing component 206.

[0024] During operation, application 202 uses discovery component 204, which implements a discovery protocol (such as Bonjour™) to identify available printers and to facilitate selecting one of the identified printers. Note that discovery component 204 communicates with a corresponding discovery component 212 in printer 106 (illustrated by dashed line 230). Next, after a printer 106 is selected, application 202 uses page-description-language component 205 to render a print job to produce corresponding printer data 208. Printer data 208 is then sent to network-printing component 206, which implements a network protocol for remote printing, such as an Internet Printing Protocol (IPP). Network-printing component 206 communicates printer data 208 to a corresponding network-printing component 214 within printer 106 (illustrated by dashed line 232).

[0025] As mentioned above, printer 106 contains a discovery component 212 and a network-printing component 214. In addition, printer 106 includes components that support printing using, for example, the JPEG (Joint Photographic Experts Group) standard 216, the PDF (Portable Document Format) standard 217, and a RF (Raster Format) standard 218. These components send data through print controller 219 to print heads 220 within printer 106, so that print controller 219 can translate the data and print heads 220 can print the translated data onto some type of print media, such as paper.

Printing Process

[0026] FIG. 3 presents a flow chart illustrating the printing process in accordance with the disclosed embodiments. First, the system uses a discovery protocol, such as the Bonjour™ protocol, to identify printers that can be accessed by computing device 102 (step 302). As mentioned above, this discovery protocol also obtains capability information from the printer. Next, the system presents a list of available printers to a user (or an application) and allows the user (or the application) to select one of the identified printers (step 304).

[0027] Next, the system checks interoperability of the selected printer (step 306). For example, the system can obtain information from the printer indicating that the printer supports JPEG, PDF and RF, in which case the system can use any of the supported formats. On the other hand, if the information indicates the printer can only support RF, the system uses RF to send data to the printer.

[0028] Next, the system has a number of options. If the selected printer supports driverless printing, the system can obtain printer capability information from the selected printer and can generate printer data for the printer based on the obtained printer capability information. Next, the system can send the generated printer data to the selected printer (step 308). As mentioned above, this capability information can be obtained from the selected printer during a query for the discovery protocol, or during a subsequent network-printing protocol query. On the other hand, if the mobile device contains a specific driver for the selected printer, the system can use the specific driver to generate and send printer data to the selected printer (step 310). The system can alternatively send the print job to a cloud comprising one or more servers that provide a printing service. This enables the cloud to generate the printer data for the selected printer (step 312).

[0029] In general, the system can select among driv-

erless printing, cloud printing and using a local driver based on a number of factors, such as power consumption and/or computational load. In one embodiment, the system uses driverless printing if possible, and if driverless printing is not supported, the system uses a local driver for the printer. Finally, if such a local driver is not supported, the system uses the cloud to generate the printer data.

Printer Capability Information

[0030] As mentioned above, the printer capability information can be obtained from the selected printer during a query for the discovery protocol (Bonjour™), or during a subsequent network-printing protocol (IPP) query. More specifically, during the discovery protocol, the selected printer can return printer-specific information specifying what types of data the selected printer can accept and information specifying the selected printer's capabilities. Moreover, this printer-specific information enables the computer device to generate the printer data for the selected printer without the need for the computer to maintain printer-specific software or printer-specific configuration information for the selected printer. Once the mobile device is actually communicating with the printer through IPP, the IPP protocol can provide the same printer-specific information. (This prevents having to cache information between the discovery protocol and the network-printing protocol.)

[0031] This printer capability information can specify attributes of the print media, such as media sizes, types (e.g., paper or photo media) and margins. This printer capability information can also specify finishing attributes, such as attributes related to stapling, hole punching and booklets. The printer capability information can additionally specify information related to printer features, such as whether it can print on two sides of a page (duplex), which output bit to use, and which media source (tray) to use. The printer capability attributes can also specify file-related attributes, such as the file sizes supported by the printer, the file-format versions supported by the printer, and the file-format extensions supported by the printer. The printer capability information can also specify various printer capabilities, such as the color spaces supported by the printer, the bit depths supported by the printer, and the resolutions supported by the printer.

[0032] This printer capability information can additionally specify current status information obtained from printers. For example, this current status information can specify: whether the printer is off-line; whether the printer is busy; or whether an error condition exists in the printer. This current status information can be presented to the user while the user is selecting an available printer.

Driverless Printing

[0033] FIG. 4 presents a flow chart illustrating the driverless printing process in accordance with the disclosed embodiments. (This flow chart illustrates some of the operations which take place during the driverless printing process which occurs in step 308 of FIG. 3.) First, the system obtains capability information for the selected printer (step 402). As mentioned above, this capability information can be obtained from the printer during a query which is part of the initial discovery protocol, or during a subsequent IPP query. Next, the system queues to the selected printer (step 404) and allows a user (or an application) to select a specific type of media for the print job (step 406).

[0034] The system then generates printer data for the selected printer based on the obtained capability information (step 408). Next, the printer sends the printer data to the selected printer (step 410). Finally, the system receives status information for the selected printer, which indicates whether the printer data was successfully printed (step 412).

Printing Process on the Printer Side

[0035] FIG. 5 presents a flow chart illustrating the printing process from the printer's perspective in accordance with the disclosed embodiments. First, the printer uses a discovery component within the printer to communicate with a client to facilitate selection of the printer by the client. During this discovery process, the discovery component can provide capability information for the printer to the client (step 502). Next, the printer uses a transport component within the printer to receive printer data from the client. During this process, the transport component can also provide capability information for the printer to the client (step 504). Finally, the printer can use a page-description-language component within the printer to print the printer data received from the client (step 506).

Document Format Preferred

[0036] To facilitate driverless printing, some of the disclosed embodiments have added new keys to the Internet Printing Protocol (IPP) standard. In particular, some embodiments include a new document-format-preferred key (as a MIME media type), which enables the printer to specify a "preferred" document format out of all of the document formats that are supported by the printer. This preferred document format can be used to improve performance. For example, the preferred document format may be more efficient to print than other document formats supported by the printer.

[0037] Note that the IPP standard already provides a document-format-supported key which specifies that a specific document format is supported by a printer. However, the existing IPP standard does not provide any way to indicate which of the supported document formats is preferred.

URF-Supported Key

[0038] Some embodiments have also added a new "URF-supported key" to a discovery protocol and a transport protocol. More specifically, some embodiments have added a new URF-supported key to the discovery protocol as part of a Bonjour™ TXT record, and have also added an analogous URF-supported key to the transport protocol as a new printer description attribute for the IPP protocol.

[0039] This URF-supported key specifies a standardized set of capabilities that are supported by a printer which enables a client to generate printer data for the client without the need for the client to maintain printer-specific software or printer-specific configuration information for the selected printer. This standardized set of capabilities is selected to enable the client to generate printer data for any type of printer.

[0040] In one embodiment, the standardized set of capabilities includes the following which also appear in the diagram of the URF-supported key 600 in FIG. 6,

(1) color spaces that are supported by the printer;
(2) bit depths that are supported by the printer for specific color spaces;
(3) a maximum number of copies supported by the printer;
(4) whether duplex printing is supported by the printer;
(5) specific finishings that are supported by the printer (e.g., stapling, folding, hole punching);
(6) input slots supported by the printer;
(7) face-up/face-down input orientation;
(8) media types supported by the printer (e.g., plain paper, glossy);
(9) output bins supported by the printer;
(10) face-up/face-down output orientation;
(11) supported print qualities; and
(12) supported resolutions.

In additional embodiments, the URF-supported key can include a subset of these capabilities.

Device-Independent Bitmap Container

[0041] Some embodiments support a new device-independent bitmap container for printer data. For example, the device-independent bitmap container can be implemented as a Multipurpose Internet Mail Extensions (MIME) subtype.

[0042] This new device-independent bitmap container includes a file header, and at least one set of a page header and a page bitmap. In one embodiment, the page header can have a predetermined format that specifies the following attributes for a page to be printed by a printer. (These attributes also appear in the device-independent bitmap container 700 which is illustrated in FIG. 7.)

(1) a bit depth;
(2) a color space;
(3) a duplex mode;
(4) a print quality;
(5) a media type;
(6) an input slot;
(7) an output bin
(8) a number of copies;
(9) one or more finishings;
(10) a width;
(11) a height; and
(12) a resolution.

In alternative embodiments, the page header can have a predetermined format that specifies a subset of these attributes.

Optimizations

[0043] In some embodiments, the system can perform optimizations to improve speed, increase print quality and save battery power. This can be accomplished by selecting a file type for the printer data that reduces the number of computational operations involved in generating the printer data, thereby improving speed and reducing the amount of power consumed by the computing device. For example, if a printer supports PDF and the computing device is printing a PDF file, it uses much less battery power and is faster to send the PDF file to the printer instead of converting the PDF file into raster data and sending the raster data to the printer. (Note that, by saving computational operations and computational time, the system frees up resources and time to perform other operations, for example to increase print quality.) Also, in the case where the cloud returns the generated printer data to the computing device, the system can select a file format which reduces the size of the printer data file. This reduces the number of data transfer operations required to forward the printer data to the printer, and thereby improves speed and reduces the amount of power consumed by the computing device.

[0044] In another example, the system can save power by selecting between driverless printing and cloud printing based on whether or not the power consumed while transferring data to and from the cloud will be offset by the power saved by off-loading the printer-related rendering operations to the cloud. By using a cloud, power-consuming computational operations can be off-loaded from a mobile device (that runs on battery power) to a server within a cloud (that runs on wall power), which can potentially save a significant amount of battery power.

Defect Solutions

[0045] In some cases, a printer manufacturer may attempt to implement a printer which adheres to the driverless printer specification, but the implementation may have one or more bugs. In this case, the system can

maintain a database containing such known bugs for specific printer models. During a printing operation, the system can first perform a lookup in the database, and if one or more known bugs exist for a printer, the system can adjust how the printer data is generated to compensate for these known bugs.

Application Programming Interfaces

[0046] One or more Application Programming Interfaces (APIs) may be used in some embodiments. An API is an interface implemented by a program code component (hereinafter "API-implementing component") that allows a different program code component (hereinafter "API-calling component") to access and use one or more functions, methods, procedures, data structures, classes, and/or other services provided by the API-implementing component. An API can define one or more parameters that are passed between the API-calling component and the API-implementing component.

[0047] An API allows a developer of an API-calling component (which may be a third-party developer) to leverage specified features provided by an API-implementing component. There may be one API-calling component, or there may be more than one such component. An API can be a source code interface that a computer system or program library provides in order to support requests for services from an application. An API can be specified in terms of a programming language that can be interpreted or compiled when an application is built.

[0048] In some embodiments the API-implementing component may provide more than one API, each providing a different view of or that access different aspects of the functionality implemented by the API-implementing component. In other embodiments the API-implementing component may itself call one or more other components via an underlying API, and thus be both an API-calling component and an API-implementing component.

[0049] An API defines the language and parameters that API-calling components use when accessing and using specified features of the API-implementing component. For example, an API-calling component accesses the specified features of the API-implementing component through one or more API calls or invocations (embodied, for example, by function or method calls) exposed by the API, and passes data and control information using parameters via the API calls or invocations. The API-implementing component may return a value through the APT in response to an API call from an API-calling component. While the API defines the syntax and result of an API call (e.g., how to invoke the API call and what the API call does), the API may not reveal how the API call accomplishes the function specified by the API call. Various API calls are transferred via the one or more application programming interfaces between the calling (API-calling component) and an API-implementing component. Transferring the API calls may include issuing, initiating, invoking, calling, receiving, returning, or responding to the function calls or messages. The function calls or other invocations of the API may send or receive one or more parameters through a parameter list or other structure. A parameter can be a constant, key, data structure, object, object class, variable, data type, pointer, array, list or a pointer to a function or method or another way to reference a data or other item to be passed via the API.

[0050] Furthermore, data types or classes may be provided by the API and implemented by the API-implementing component. Thus, the API-calling component may declare variables, use pointers to, use or instantiate constant values of such types or classes by using definitions provided in the API.

[0051] Generally, an API can be used to access a service or data provided by the API-implementing component, or to initiate performance of an operation or computation provided by the API-implementing component. By way of example, the API-implementing component and the API-calling component may be an operating system, a library, a device driver, an API, an application program, or other module (it should be understood that the API-implementing component and the API-calling component may be the same or different type of module from each other). API-implementing components may in some cases be embodied at least in part in firmware, microcode, or other hardware logic. In some embodiments, an API may allow a client program to use the services provided by a Software Development Kit (SDK) library. In other embodiments an application or other client program may use an API provided by an Application Framework.

[0052] In these embodiments the application or client program may incorporate calls to functions or methods provided by the SDK and provided by the API, or use data types or objects defined in the SDK and provided by the API. An application framework may in these embodiments provide a main event loop for a program that responds to various events defined by the framework. The API allows the application to specify the events and the responses to the events using the application framework. In some implementations, an API call can report to an application the capabilities or state of a hardware device, including those related to aspects such as input capabilities and state, output capabilities and state, processing capability, power state, storage capacity and state, communications capability, etc., and the API may be implemented in part by firmware, microcode, or other low-level logic that executes in part on the hardware component.

[0053] The API-calling component may be a local component (i.e., on the same data processing system as the API-implementing component) or a remote component (i.e., on a different data processing system from the API-implementing component) that communicates with the API-implementing component through the API over a network. It should be understood that an API-implementing component may also act as an API-calling component (i.e., it may make API calls to an API exposed by a dif-

ferent API-implementing component), and an API-calling component may also act as an API-implementing component by implementing an API that is exposed to a different API-calling component.

[0054] The API may allow multiple API-calling components written in different programming languages to communicate with the API-implementing component (thus, the API may include features for translating calls and returns between the API-implementing component and the API-calling component); however, the API may be also implemented in terms of a specific programming language.

[0055] FIG. 8 is a block diagram illustrating an exemplary API architecture, which may be used in some embodiments of the invention. As shown in FIG. 8, the API architecture 800 includes the API-implementing component 810 (e.g., an operating system, a library, a device driver, an API, an application program, or other module) that implements the API 820. The API 820 specifies one or more functions, methods, classes, objects, protocols, data structures, formats and/or other features of the API-implementing component that may be used by the API-calling component 830. The API 820 can specify at least one calling convention that specifies how a function in the API-implementing component receives parameters from the API-calling component and how the function returns a result to the API-calling component. The API-calling component 830 (e.g., an operating system, a library, a device driver, an API, an application program, or other module) makes API calls through the API 820 to access and use the features of the API-implementing component 810 that are specified by the API 820. The API-implementing component 810 may return a value through the API 820 to the API-calling component 830 in response to an API call.

[0056] It will be appreciated that the API-implementing component 810 may include additional functions, methods, classes, data structures, and/or other features that are not specified through the API 820 and are not available to the API-calling component 830. It should be understood that the API-calling component 830 may be on the same system as the API-implementing component 810 or may be located remotely and access the API-implementing component 810 using the API 820 over a network. While FIG. 8 illustrates a single API-calling component 830 interacting with the API 820, it should be understood that other API-calling components, which may be written in different languages (or the same language) than the API-calling component 830, may use the API 820.

[0057] The API-implementing component 810, the API 820, and the API-calling component 830 may be stored in a machine-readable medium, which includes any mechanism for storing information in a form readable by a machine (e.g., a computer or other data processing system). For example, a machine-readable medium includes magnetic disks, optical disks, random access memory, read only memory, flash memory devices, etc.

[0058] In FIG. 9 ("Software Stack"), an exemplary embodiment, applications can make calls to Services A or B using Service API and to Operating System (OS) using OS API. Services A and B can make calls to OS using OS API.

[0059] The foregoing descriptions of embodiments have been presented for purposes of illustration and description only. They are not intended to be exhaustive or to limit the present description to the forms disclosed. Accordingly, many modifications and variations will be apparent to practitioners skilled in the art. Additionally, the above disclosure is not intended to limit the present description. The scope of the present description is defined by the appended claims.

ADDITIONAL STATEMENTS

[0060] There is provided a system that facilitates driverless printing, comprising: a discovery component configured to identify accessible printers and to facilitate selection of an accessible printer, wherein the discovery component is also configured to obtain capability information from accessible printers; a transport component configured to transport data to the selected printer; and a page-description-language component configured to generate printer data for the selected printer based on obtained capability information for the selected printer; wherein the capability information obtained from the selected printer enables the system to generate the printer data for the selected printer without the need for the system to maintain printer-specific software or printer-specific configuration information for the selected printer.

[0061] Preferably, wherein the system provides an integrated framework that enables the discovery component, the transport component and the page-description-language component to be accessed by an application.

[0062] Preferably, wherein the integrated framework includes one or more application programming interfaces (APIs) that provide access to the discovery component, the transport component and the page-description-language component.

[0063] Preferably, wherein while obtaining the capability information from accessible printers, the discovery component is configured to obtain current status information from the accessible printers.

[0064] Preferably, wherein the current status information specifies one or more of the following: whether the printer is off-line; whether the printer is busy; and whether an error condition exists in the printer.

[0065] Preferably, wherein the capability information for a printer specifies one or more of the following: resolutions; color spaces; bit depths; input slots; face-up/face-down input orientation; output bins; face-up/face-down output orientation; duplex printing support; media types; copy support; supported finishings; and print quality.

[0066] Preferably, wherein the capability information for the printer additionally specifies one or more of the

following: file sizes; file-format versions; and file-format extensions.

[0067] Preferably, wherein the transport component is also configured to obtain capability information from the selected printer.

[0068] There is provided a computing device that facilitates driverless printing, comprising: a discovery component within the computing device configured to identify accessible printers and to facilitate selection of an accessible printer, wherein the discovery component is also configured to obtain capability information from accessible printers; a transport component within the computing device configured to transport data to the selected printer; and a page-description-language component within the computing device configured to generate the printer data for the selected printer based on obtained capability information for the selected printer; wherein the capability information obtained from the selected printer enables the computing device to generate the printer data for the selected printer without the need for the computing device to maintain printer-specific software or printer-specific configuration information for the selected printer.

[0069] Preferably, wherein while obtaining the capability information from accessible printers, the discovery component is configured to obtain current status information from the accessible printers.

[0070] Preferably, wherein the transport component is also configured to obtain capability information from the selected printer.

[0071] There is provided a method for facilitating driverless printing, comprising: using a discovery component to identify accessible printers and to facilitate selection of an accessible printer, wherein the discovery component is also configured to obtain capability information from accessible printers; using a transport component to transport data to the selected printer; and using a page-description-language component to generate printer data for the selected printer based on obtained capability information for the selected printer; wherein the capability information obtained from the selected printer enables the method to generate the printer data for the selected printer without the need for a computing device upon which the method operates to maintain printer-specific software or printer-specific configuration information for the selected printer.

[0072] Preferably, wherein the discovery component, the transport component and the page-description-language component are accessed through a unified framework which can be accessed by an application.

[0073] Preferably, wherein the integrated framework includes one or more application programming interfaces (APIs) that provide access to the discovery component, the transport component and the page-description-language component.

[0074] Preferably, wherein while obtaining the capability information from accessible printers, the discovery component is configured to obtain current status information from the accessible printers.

[0075] Preferably, wherein the current status information specifies one or more of the following: whether the printer is off-line; whether the printer is busy; and whether an error condition exists in the printer.

[0076] Preferably, wherein the capability information for a printer specifies one or more of the following: resolutions; color spaces; bit depths; input slots; face-up/face-down input orientation; output bins; face-up/face-down output orientation; duplex printing support; media types; copy support; supported finishings; and print quality.

[0077] Preferably, wherein the capability information for the printer additionally specifies one or more of the following: file sizes; file-format versions; and file-format extensions.

[0078] Preferably, wherein the transport component is also configured to obtain capability information from the selected printer.

[0079] There is provided a computer-readable storage medium storing instructions that when executed by a computer cause the computer to provide a system that facilitates driverless printing, wherein the instructions implement a number of components, including: a discovery component configured to identify accessible printers and to facilitate selection of an accessible printer, wherein the discovery component is also configured to obtain capability information from accessible printers; a transport component configured to transport data to the selected printer; and a page-description-language component configured to generate printer data for the selected printer based on obtained capability information for the selected printer; wherein the capability information obtained from the selected printer enables the system to generate the printer data for the selected printer without the need for the system to maintain printer-specific software or printer-specific configuration information for the selected printer.

[0080] There is provided a machine-implemented method in an environment which uses an Application Programming Interface (API), the method comprising: transferring a discovery call through the API to cause a discovery component to identify accessible printers and to facilitate selection of an accessible printer, wherein the discovery component is also configured to obtain capability information from accessible printers; transferring a transport call through the API to cause a transport component to transport data to the selected printer; and transferring a rendering call through the API to cause a page-description-language component to generate printer data for the selected printer based on obtained capability information for the selected printer; wherein the capability information obtained from the selected printer enables the system to generate the printer data for the selected printer without the need for the system to maintain printer-specific software or printer-specific configuration information for the selected printer.

[0081] Preferably, wherein while obtaining the capability information from accessible printers, the discovery

component is configured to obtain current status information from the accessible printers.

[0082] Preferably, wherein the transport component is also configured to obtain capability information from the selected printer.

[0083] There is provided a data processing system, comprising:a processor to execute instructions; and a memory coupled with the processor to store instructions which, when executed by the processor, cause the processor to: interface a discovery component of the data processing system with an Application Programming Interface (API)-calling component to identify accessible printers to facilitate selecting an accessible printer, and to obtain capability information from accessible printers; interface a transport component of the data processing system with the API-calling component to transport data to the selected printer; and interface a page-description-language component of the data processing system with the API-calling component to generate printer data for the selected printer based on obtained capability information for the selected printer; wherein the capability information obtained from the selected printer enables the processor to generate the printer data for the selected printer without the need for the data processing system to maintain printer-specific software or printer-specific configuration information for the selected printer.

Claims

1. A printer (106) that facilitates driverless printing, comprising:

a discovery component (212) configured to identify the printer (106) to a client (102), facilitate selection of the printer (106) by the client (102), obtain capability information from the printer (106), and send the capability information to the client (102);
a transport component (214) configured to receive printer data (208) from the client (102); and
a page-description-language component configured to print the received printer data (208);

wherein the capability information enables the client device (102) to generate the printer data (208) for the printer (106) without the client device (102) needing to maintain printer-specific software or printer-specific configuration information for the printer (106);
characterised by:

the capability information including identification of a preferred document format, from a plurality of document formats supported by the printer (106), wherein the preferred document format comprises a format that is more efficiently printed by the printer (106) than another of the printer's supported document formats; and
the printer data (208) being generated in the preferred document format.

2. The printer (106) of claim 1, wherein the printer (106) provides an integrated framework that enables the discovery component (212), the transport component (214), and the page-description-language component within the printer (106) to be accessed by an application (202).

3. The printer (106) of claim 2, wherein the integrated framework includes one or more application programming interfaces (APIs) (203; 820) that provide access to the discovery component (212), the transport component (214) and the page-description-language component within the printer (106).

4. The printer (106) of any preceding claim, wherein the discovery component (212) provides current status information from the printer (106) to the client (102).

5. The printer (106) of any preceding claim, wherein a cloud printing system facilitates cloud printing in lieu of driverless printing by receiving the printer data (208) from the client (102) and printing the printer data (208) in response to a determination that power consumed while transferring data to and from the cloud printing system will be offset by power saved by off-loading printer-related rendering operations to the cloud printing system.

6. The printer (106) of any preceding claim, wherein the capability information for the printer specifies one or more of the following:

resolutions;
color spaces;
bit depths;
input slots;
face-up/face-down input orientation;
output bins;
face-up/face-down output orientation;
duplex printing support;
media types;
copy support;
supported finishings; and
print quality.

7. The printer (106) of claim 6, wherein the capability information for the printer additionally specifies one or more of the following:

file sizes;
file-format versions; and
file-format extensions.

8. The printer (106) of any preceding claim, wherein the preferred document format is specified using a MIME media type.

9. A method for using a printer (106) that facilitates driverless printing, the printer (106) comprising a discovery component (212), a transport component (214), and a page-description-language component, and the method comprising:

using the discovery component (212) to identify the printer (106) to a client (102), facilitate selection of the printer (106) by the client (102), obtain capability information from the printer (106), and send the capability information to the client (102);
using the transport component (214) to receive printer data (208) from the client (102); and
using the page-description-language component to print the received printer data (208);

wherein the capability information enables the client device (102) to generate the printer data (208) for the printer (106) without the client device (102) needing to maintain printer-specific software or printer-specific configuration information for the printer (106);
the capability information including identification of a preferred document format, from a plurality of document formats supported by the printer (106), wherein the preferred document format comprises a format that is more efficiently printed by the printer (106) than another of the printer's supported document formats; and
the printer data (208) being generated in the preferred document format.

10. The method of claim 9, wherein the discovery component (212), the transport component (214), and the page-description-language component are accessed through a unified framework which can be accessed by an application, and wherein the unified framework includes one or more application programming interfaces (APIs) (203;820) that provide access to the discovery component (212), the transport component (214) and the page-description-language component within the printer (106).

11. The method of claims 9 or 10, further comprising:

using the discovery component (212) to provide current status information from the printer (106) to the client (102).

12. The method of claims 9, 10, or 11, wherein the capability information for the printer specifies one or more of the following:

resolutions;
color spaces;
bit depths;
input slots;
face-up/face-down input orientation;
output bins;
face-up/ face-down output orientation;
duplex printing support;
media types;
copy support;
supported finishings;
print quality;
file sizes;
file-format versions; and
file-format extensions.

13. The method of claims 9, 10, 11, or 12, further comprising:

using a cloud printing system to facilitate cloud printing in lieu of driverless printing, wherein the cloud printing system is to receive the printer data (208) from the client (102) and to print the printer data (208) in response to a determination that power consumed while transferring data to and from the cloud printing system will be offset by power saved by off-loading printer-related rendering operations to the cloud printing system.

14. A computer-readable storage medium storing instructions that when executed by a computer cause the computer to provide a printer (106) that facilitates driverless printing, wherein the instructions implement a number of components, including:

a discovery component (212) to identify the printer (106) to a client (102), facilitate selection of the printer (106) by the client (102), obtain capability information from the printer (106), and send the capability information to the client (102);
a transport component (214) to receive printer data (208) from the client (102); and
a page-description-language component to print the received printer data (208);

wherein the capability information enables the client device (102) to generate the printer data (208) for the printer (106) without the client device (102) needing to maintain printer-specific software or printer-specific configuration information for the printer (106);
the capability information including identification of a preferred document format, from a plurality of document formats supported by the printer (106), wherein the preferred document format comprises a format that is more efficiently printed by the printer (106)

than another of the printer's supported document formats; and
the printer data (208) being generated in the preferred document format.

15. The computer-readable storage medium of claim 14, further comprising instructions that when executed by a computer cause the computer to facilitate cloud printing in lieu of driverless printing, wherein the instructions implement a number of components, including:

a cloud printing system to receive the printer data (208) from the client (102) and to print the printer data (208), wherein the cloud printing system receives and prints the printer data (208) in response to a determination that power consumed while transferring data to and from the cloud printing system will be offset by power saved by off-loading printer-related rendering operations to the cloud printing system.

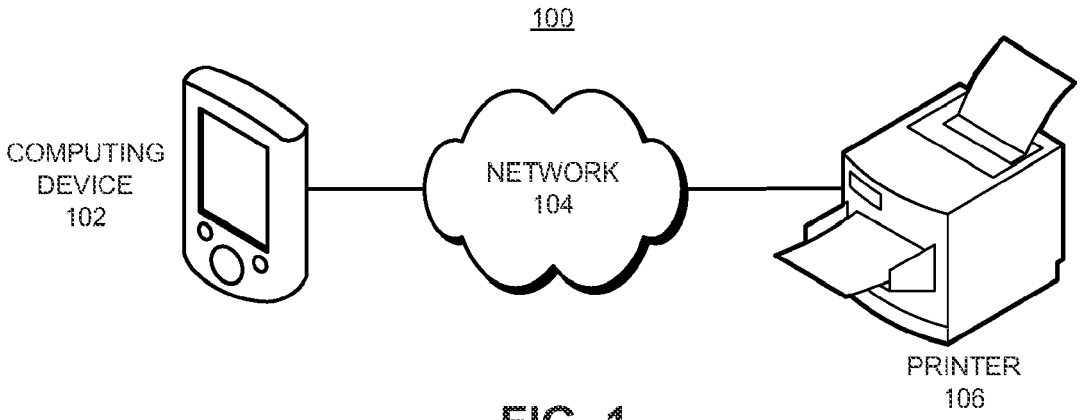

FIG. 1

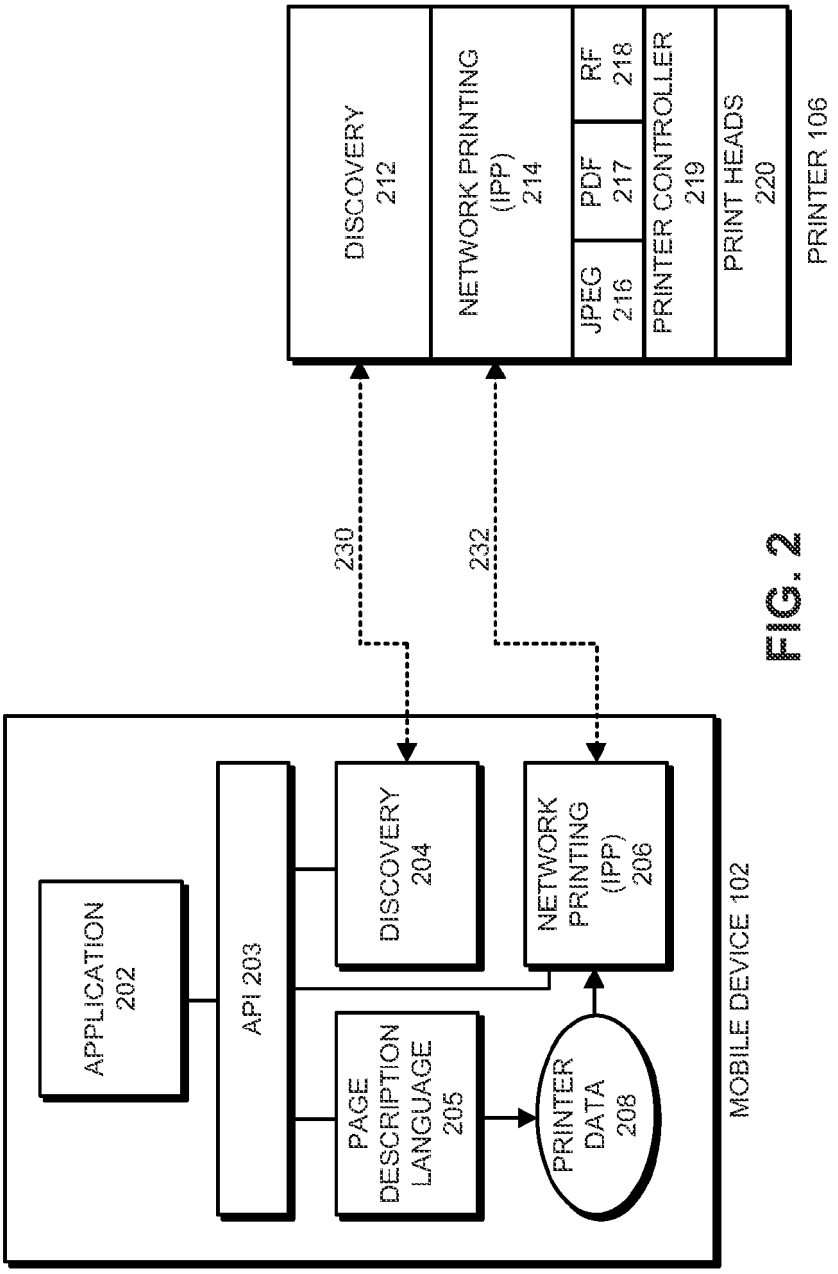
DISCOVERY 212
NETWORK PRINTING (IPP) 214
JPEG 216
PDF 217
RF 218
PRINTER CONTROLLER 219
PRINT HEADS 220
PRINTER 106
230
232
APPLICATION 202
API 203
DISCOVERY 204
NETWORK PRINTING (IPP) 206
PAGE DESCRIPTION LANGUAGE 205
PRINTER DATA 208
MOBILE DEVICE 102

FIG. 2

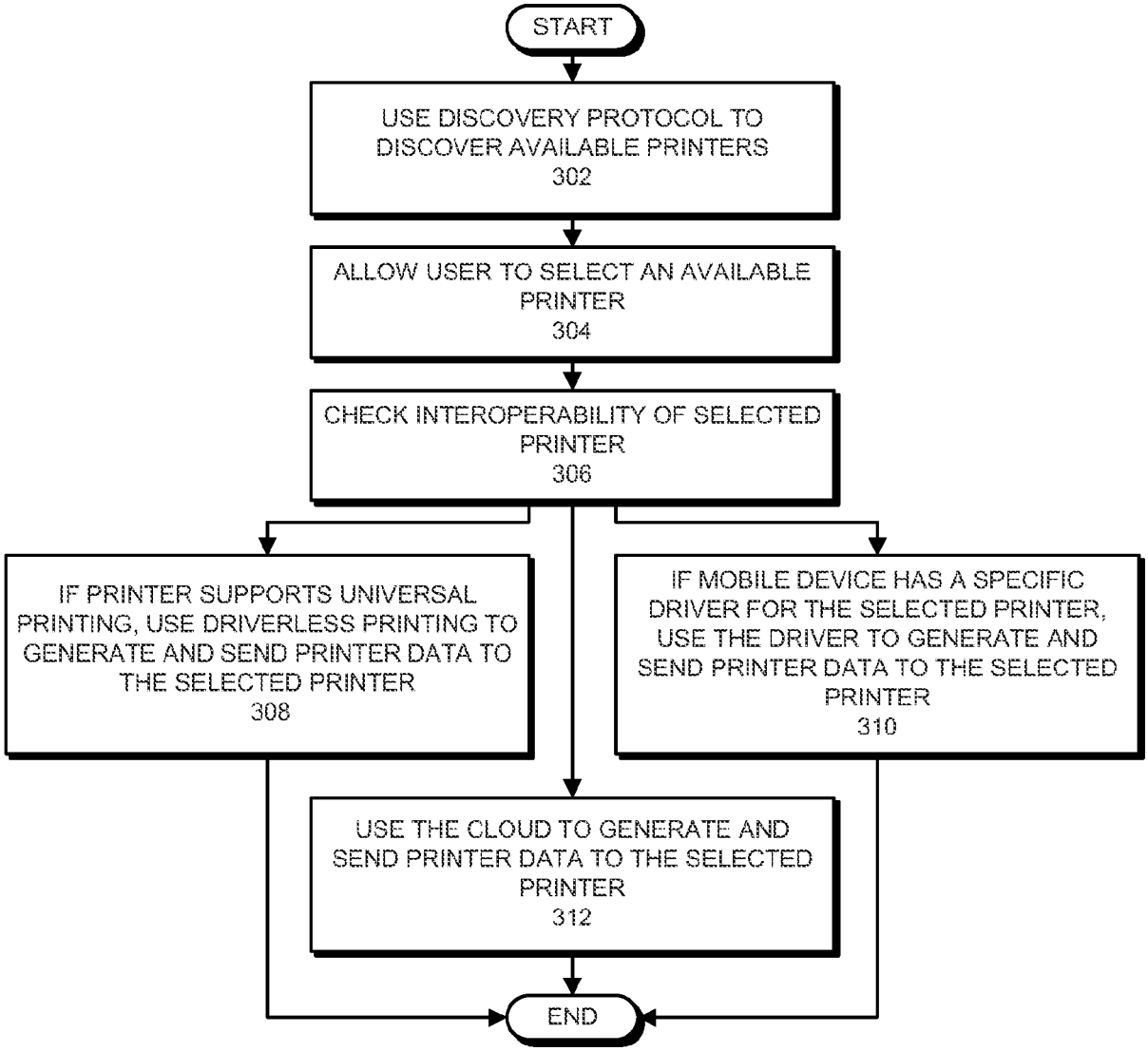
START
USE DISCOVERY PROTOCOL TO DISCOVER AVAILABLE PRINTERS
302
ALLOW USER TO SELECT AN AVAILABLE PRINTER
304
CHECK INTEROPERABILITY OF SELECTED PRINTER
306
IF PRINTER SUPPORTS UNIVERSAL PRINTING, USE DRIVERLESS PRINTING TO GENERATE AND SEND PRINTER DATA TO THE SELECTED PRINTER
308
IF MOBILE DEVICE HAS A SPECIFIC DRIVER FOR THE SELECTED PRINTER, USE THE DRIVER TO GENERATE AND SEND PRINTER DATA TO THE SELECTED PRINTER
310
USE THE CLOUD TO GENERATE AND SEND PRINTER DATA TO THE SELECTED PRINTER
312
END

FIG. 3

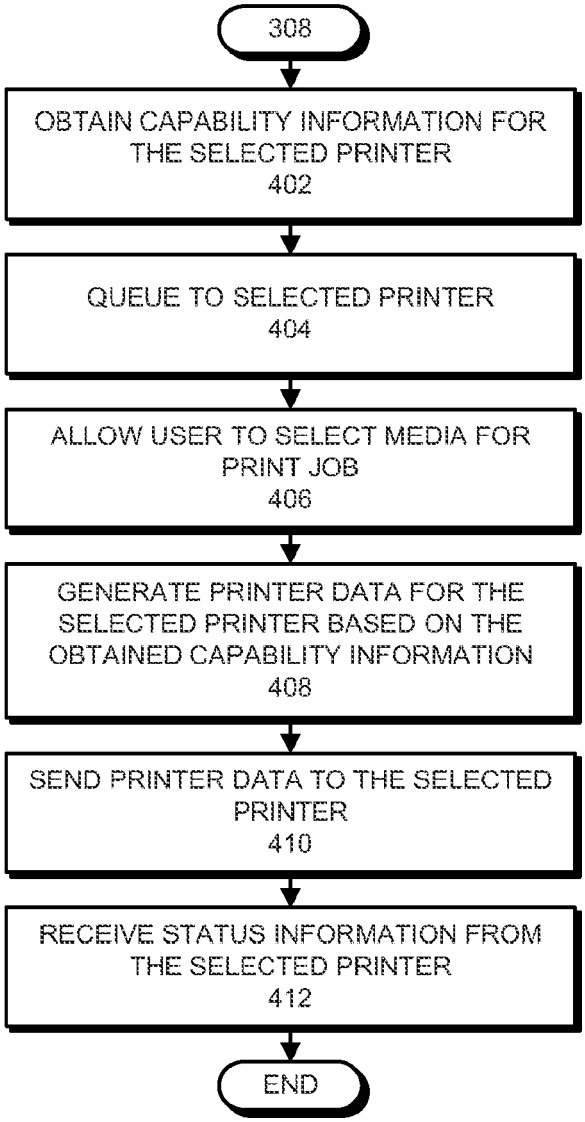
308
OBTAIN CAPABILITY INFORMATION FOR THE SELECTED PRINTER
402
QUEUE TO SELECTED PRINTER
404
ALLOW USER TO SELECT MEDIA FOR PRINT JOB
406
GENERATE PRINTER DATA FOR THE SELECTED PRINTER BASED ON THE OBTAINED CAPABILITY INFORMATION
408
SEND PRINTER DATA TO THE SELECTED PRINTER
410
RECEIVE STATUS INFORMATION FROM THE SELECTED PRINTER
412
END

FIG. 4

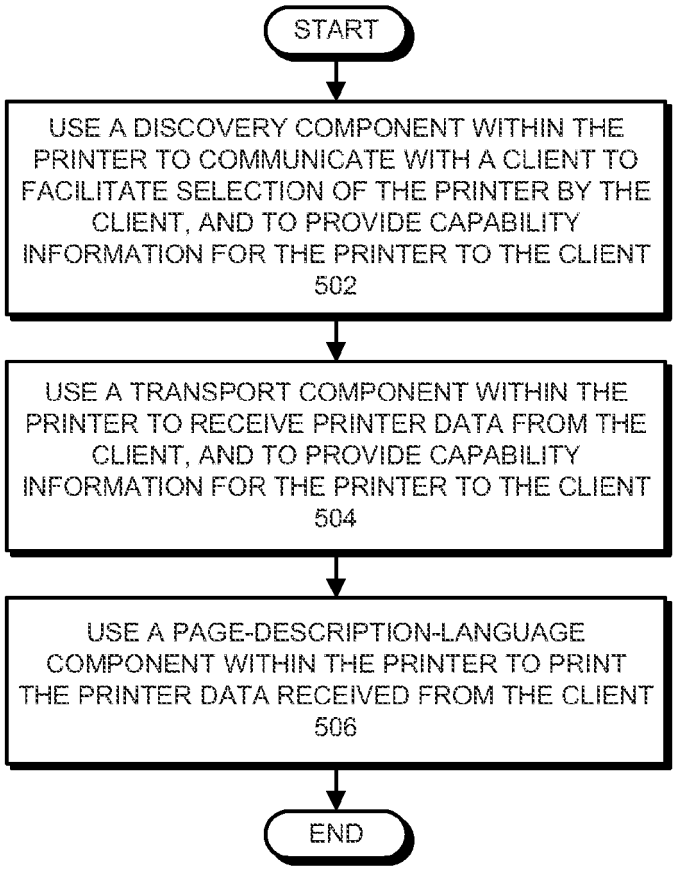
START
USE A DISCOVERY COMPONENT WITHIN THE PRINTER TO COMMUNICATE WITH A CLIENT TO FACILITATE SELECTION OF THE PRINTER BY THE CLIENT, AND TO PROVIDE CAPABILITY INFORMATION FOR THE PRINTER TO THE CLIENT
502
USE A TRANSPORT COMPONENT WITHIN THE PRINTER TO RECEIVE PRINTER DATA FROM THE CLIENT, AND TO PROVIDE CAPABILITY INFORMATION FOR THE PRINTER TO THE CLIENT
504
USE A PAGE-DESCRIPTION-LANGUAGE COMPONENT WITHIN THE PRINTER TO PRINT THE PRINTER DATA RECEIVED FROM THE CLIENT
506
END

FIG. 5

URF-SUPPORTED KEY
600

COLOR SPACES
BIT DEPTHS
MAX COPIES
DUPLEX SUPPORT
FINISHINGS
INPUT SLOTS
INPUT ORIENTATION
MEDIA TYPES
OUTPUT BINS
OUTPUT ORIENTATION
PRINT QUALITIES
RESOLUTIONS

FIG. 6

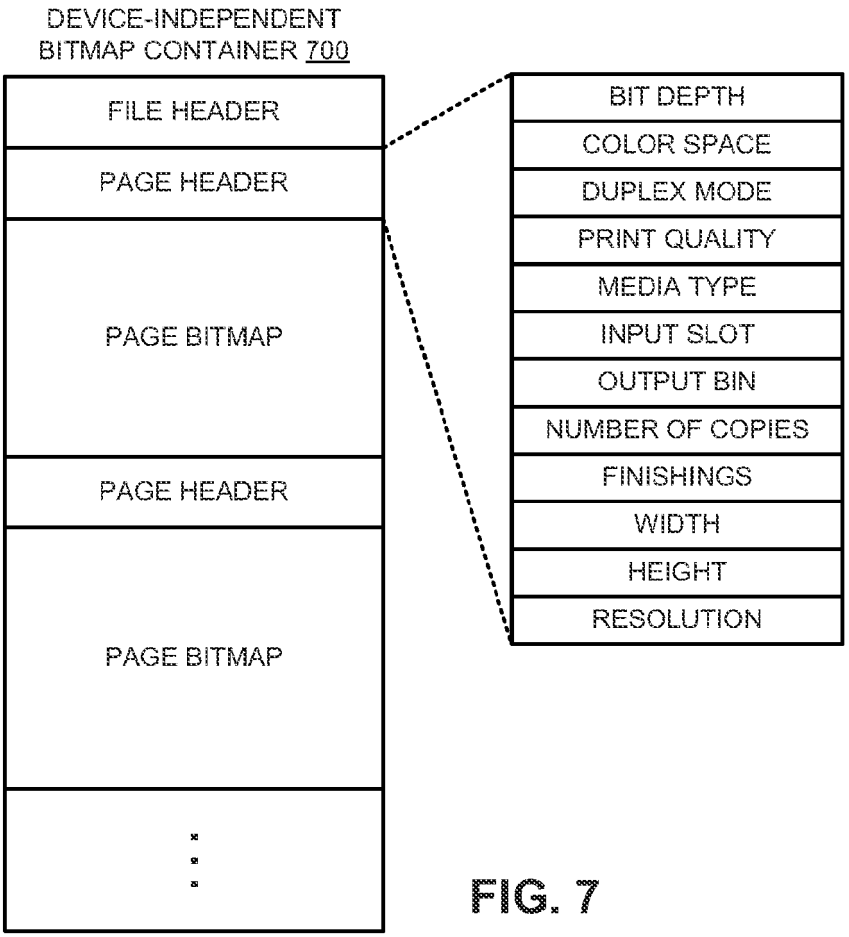
DEVICE-INDEPENDENT
BITMAP CONTAINER 700
FILE HEADER
PAGE HEADER
PAGE BITMAP
PAGE HEADER
PAGE BITMAP
BIT DEPTH
COLOR SPACE
DUPLEX MODE
PRINT QUALITY
MEDIA TYPE
INPUT SLOT
OUTPUT BIN
NUMBER OF COPIES
FINISHINGS
WIDTH
HEIGHT
RESOLUTION

FIG. 7

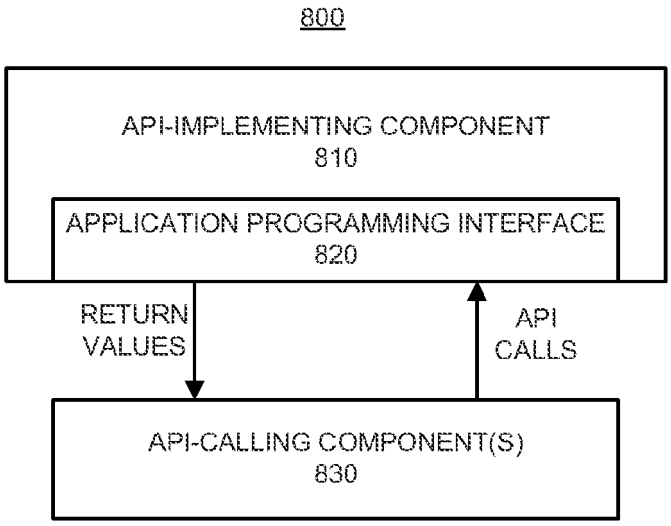
800
API-IMPLEMENTING COMPONENT
810
APPLICATION PROGRAMMING INTERFACE
820
RETURN
VALUES
API
CALLS
API-CALLING COMPONENT(S)
830

FIG. 8

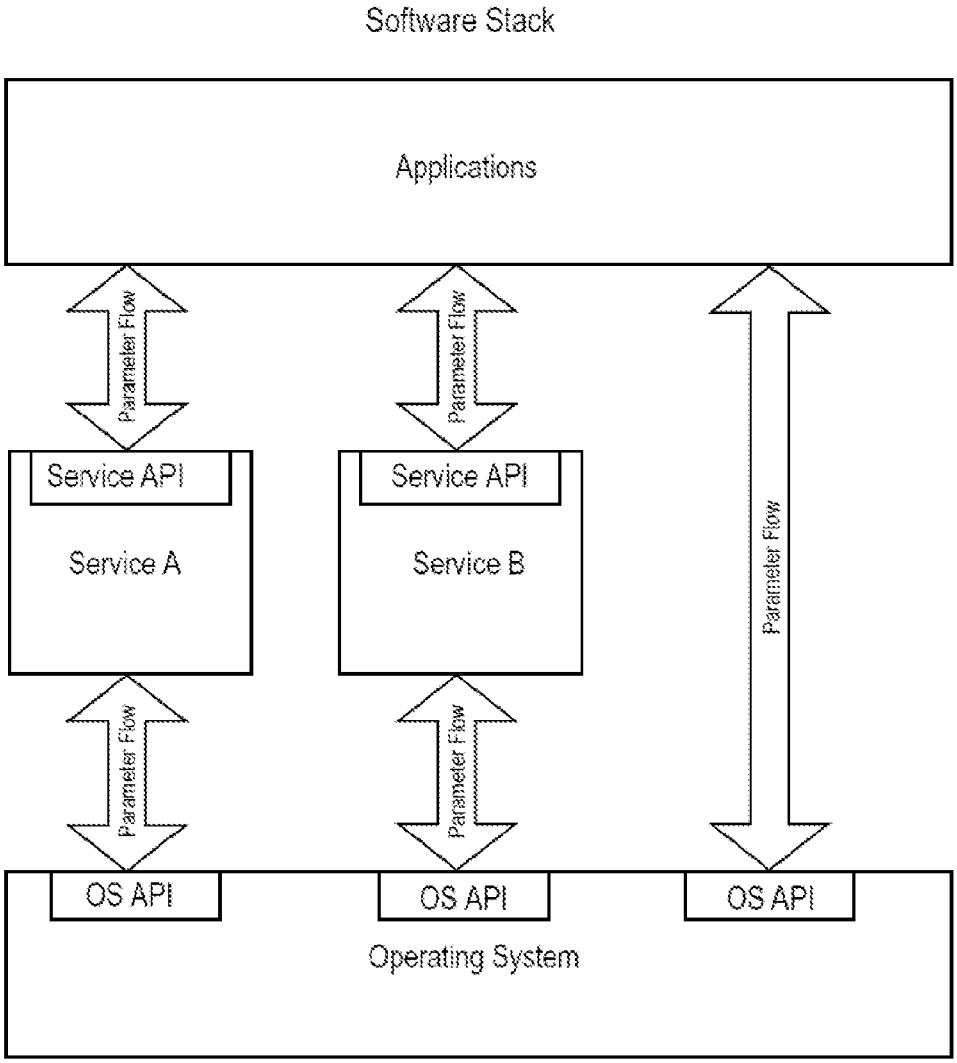
Software Stack
Applications
Parameter Flow
Parameter Flow
Parameter Flow
Service API
Service A
Service API
Service B
Parameter Flow
Parameter Flow
Parameter Flow
OS API
OS API
OS API
Operating System

FIG. 9

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 16 18 7920

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	EP 1 450 515 A2 (HEWLETT PACKARD DEVELOPMENT CO [US]) 25 August 2004 (2004-08-25) * paragraph [0031] - paragraph [0115] * * figures 10, 14 *	1-15	INV. G06F3/12
A	US 2007/226238 A1 (KIILERICH DENNIS A [US] ET AL) 27 September 2007 (2007-09-27) * paragraph [0016] - paragraph [0020] * * figure 4 *	1-15	

TECHNICAL FIELDS SEARCHED (IPC)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
The Hague	30 November 2016	Kochev, Miroslav

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 16 18 7920

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

30-11-2016

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1450515 A2	25-08-2004	EP 1450515 A2	25-08-2004
		JP 2004252979 A	09-09-2004
		KR 20040074960 A	26-08-2004
		US 2004160623 A1	19-08-2004
US 2007226238 A1	27-09-2007	NONE	

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82